MCR Key Logger Preliminary Design

Beau Harrison & Bobby Santucci

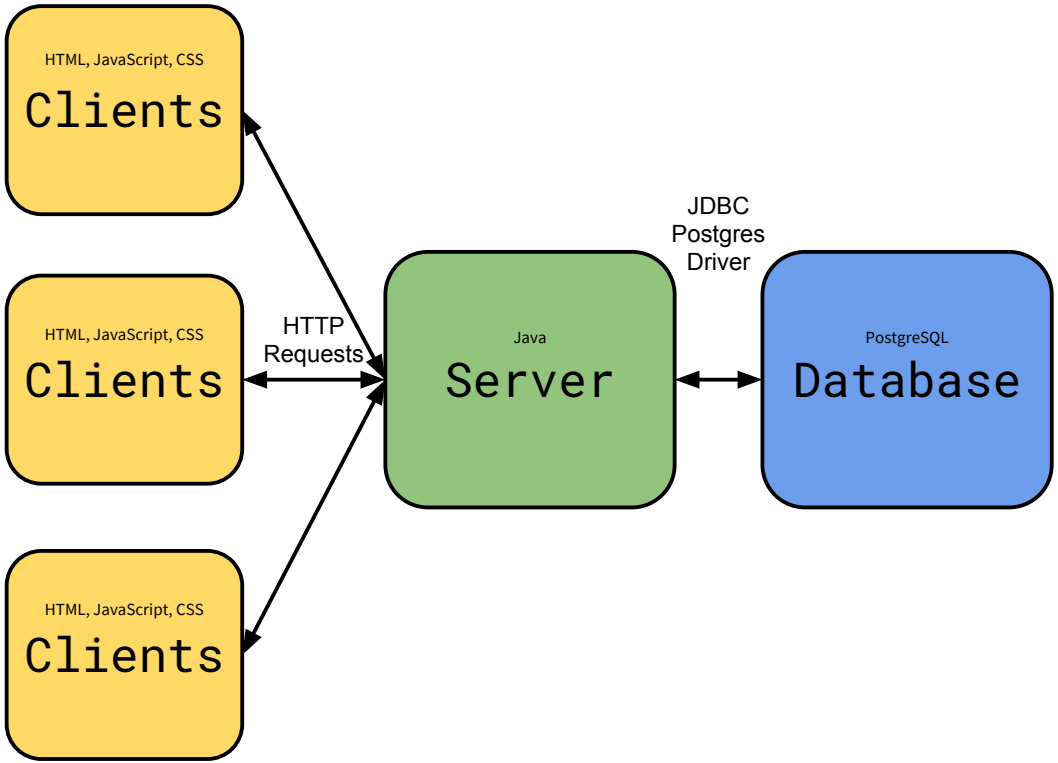
15 October 2018

Goals

Minimally viable product

- The system will
  - Be supported and owned by Operations
    - 24/7
  - Use modern tools known by the lab
  - Replace current system
    - No loss of data
  - Allow for future expansion
- The user interface will
  - Take a key and id
    - Check-out a key to a qualified employee
    - Check-in a key
  - Display currently checked-out keys to
    - Sort
    - Filter
  - Display all transactions to
    - Sort
    - Filter

Basic Structure



Server

- Tomcat
  - Defines end-points
  - Initiates SQL queries
  - Easy to maintain and develop
  - Known and used in Controls
  - Security is built in

Database

- Currently FileMaker Pro
- Oracle/Postgres/MariaDB are lab supported
- We chose Postgres
  - Stored procedures reduce network connections
  - Validation
  - Action
- Replication (Hardware dependent)
  - Seamless fail over
  - Invisible to users
  - Backup

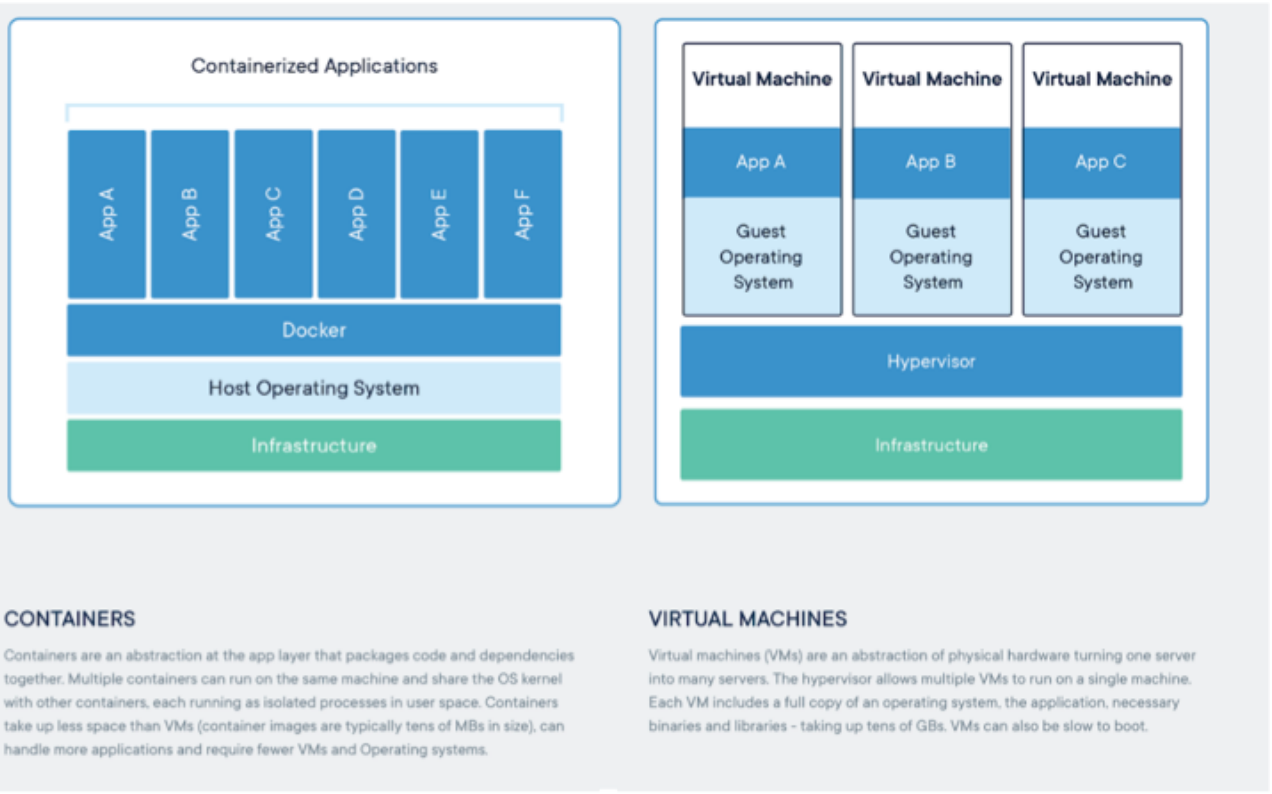
Client

- Web page
  - Easily distributed
  - Nothing to install
- Single page application
  - Simplicity
  - Majority of actions take place in a single view
- React
  - Provides an interface for custom elements
  - Limits DOM update (speed)

Deployment/Development/Production

- Docker
  - “Build once, run anywhere”
  - Light weight for fast start up
  - Existing containers
  - Modularity and scalability

Deployment/Development/Production



Fermilab

Outstanding issues

- Security
  - IP restricted currently
  - The future is SSO
- User prompts
  - Seven outstanding issues
- Admin interface
  - Necessary?
- Maintenance
  - Upgrades
  - CRON
  - Postgres

Options explored

- Computing Division services
  - Central web services
  - Database hosting
  - Hesitation with regards to criticality and uptime requests
    - Necessary?
  - Network reliability
- Operations
  - Tandem laptop
    - No network issues
    - Not robust
  - MCR server hardware
    - Wilson Hall power
    - Local network
      - Decided to replicate training database locally

Lessons learned

- Testing!
  - Development was different from production
  - Changes are unreliable
- Structure
  - Anytime we veered from “rules” things got more difficult or confusing
- Rely on existing expertise
  - Get to solutions quicker

Plans for the future

- Native mobile apps
- RFID keys
- Electronic RWPs (ESH&Q)
  - Integration with electronic worklists
- Keyosks (tablets) for self check-out
- Read only access via on-site access
- Database replication
- Documentation!